Query Optimization

CPS 216

Advanced Database Systems

Correlated subqueries

- SELECT CID FROM Course
  WHERE title LIKE 'CPS%'
  AND min_enroll > (SELECT COUNT(*) FROM Enroll
  WHERE Enroll.CID = Course.CID);
- Executing correlated subquery is expensive
  - The subquery is evaluated once for every CPS course

➢ Decorrelate!

2

COUNT bug

- SELECT CID FROM Course
  WHERE title LIKE 'CPS%'
  AND min_enroll > (SELECT COUNT(*) FROM Enroll
  WHERE Enroll.CID = Course.CID);
- SELECT CID First compute the enrollment for all(?) courses
  FROM Course, (SELECT CID, COUNT(*) AS cnt
  FROM Enroll GROUP BY CID) t
  WHERE t.CID = Course.CID AND min_enroll > t.cnt
  AND title LIKE 'CPS%';
- Suppose a CPS class is empty
  - The first query returns this course; the second does not

3

Magic decorrelation

- Simple idea
  - Process the outer query using other predicates
    - To collect bindings for correlated variables in the subquery
  - Evaluate the subquery using the bindings collected
    - It is a join
    - Once for the entire set of bindings
      - Compared to once per binding in the naïve approach
  - Use the result of the subquery to refine the outer query
    - Another join
- Name "magic" comes from a technique in recursive processing of Datalog queries

4

Magic example (slide 1)

- Original query
  - SELECT CID FROM Course
    WHERE title LIKE 'CPS%'
    AND min_enroll > (SELECT COUNT(*) FROM Enroll
    WHERE Enroll.CID = Course.CID);
- Process the outer query without the subquery
  - CREATE VIEW Supp_Course AS
    SELECT * FROM Course WHERE title LIKE 'CPS%';
- Collect bindings
  - CREATE VIEW Magic AS
    SELECT DISTINCT CID FROM Supp_Course;

5

Magic example (slide 2)

- Evaluate the subquery with bindings
  - CREATE VIEW DS AS
    SELECT Enroll.CID, COUNT(*) AS cnt
    FROM Magic, Enroll WHERE Magic.CID = Enroll.CID
    GROUP BY Enroll.CID;
    UNION
    SELECT Magic.CID, 0 AS cnt -- the COUNT patch
    FROM Magic
    WHERE Magic.CID NOT IN (SELECT CID FROM Enroll);
- Finally, refine the outer query
  - SELECT Supp_Course.CID FROM Supp_Course, DS
    WHERE Supp_Course.CID = DS.CID
    AND min_enroll > DS.cnt;

6

Summary of query rewrite

- Break the artificial boundary between queries and subqueries
- Combine as many query blocks as possible in a select-project-join block, where the clean rules of relational algebra apply
- Handle with care—extremely tricky with duplicates, NULL's, empty tables, and correlation

7

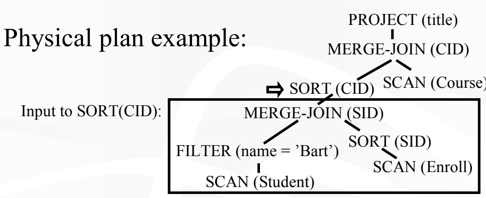
Review of the bigger picture

- Heuristics-based optimization
  – Apply heuristics to rewrite plans into cheaper ones
- Cost-based optimization
  – Rewrite logical plan to combine blocks as much as possible
  – Optimize query block by block
    - Enumerate logical and physical plans (Thursday)
    - Estimate the cost of plans (today)
    - Pick a plan with acceptable cost (Thursday)
  – Focus: select-project-join blocks

8

Cost estimation

Physical plan example:



- We have: cost estimation for each operator
  – Example: SORT(CID) takes $2 \times B(input)$
    - But what is $B(input)$?
- We need: size of intermediate results

9

Selections with equality predicates

- Q: $\sigma_{A=v} R$
- Suppose the following information is available
  – Size of R: $|R|$
  – Number of distinct A values in R: $|\pi_A R|$
- Assumptions
  – Values of A are uniformly distributed in R
  – Values of v in Q are uniformed distributed over all $R.A$ values
- $|Q| \approx |R| / |\pi_A R|$
  – Selectivity factor of $A = v$ is $1 / |\pi_A R|$

10

Conjunctive predicates

- Q: $\sigma_{A=u \text{ AND } B=v} R$
- Additional assumptions
  – $A = u$ and $B = v$ are independent
    - Counterexample: major and advisor
  – No "over"-selection
    - Counterexample: A is the key
- $|Q| \approx |R| / (|\pi_A R| \cdot |\pi_B R|)$
  – Reduce total size by all selectivity factors

11

Negated and disjunctive predicates

- Q: $\sigma_{A<>v} R$
  – $|Q| \approx |R| \cdot (1 - 1/|\pi_A R|)$
    - Selectivity factor of $\neg p$ is (1 – selectivity factor of p)
- Q: $\sigma_{A=u \text{ OR } B=v} R$
  – $|Q| \approx |R| \cdot (1/|\pi_A R| + 1/|\pi_B R|)$?
    - No! Tuples satisfying $A = u$ AND $B = v$ are counted twice
  – $|Q| \approx |R| \cdot (1 - (1 - 1/|\pi_A R|) \cdot (1 - 1/|\pi_B R|))$
    - Intuition: $A = u$ OR $B = v$ is equivalent to $\neg(\neg(A = u) \text{ AND } \neg(B = v))$

12

Range predicates

- Q: $\sigma_{A > v} R$
- Not enough information!
  - Just pick $| Q | = | R | \cdot 1/3$
- With more information
  - Largest $R.A$ value: high($R.A$)
  - Smallest $R.A$ value: low($R.A$)
  - $| Q | = | R | \cdot (\text{high}(R.A) - v) / (\text{high}(R.A) - \text{low}(R.A))$
  - In practice: sometimes the second highest and lowest are used instead
    - The highest and the lowest are often used by inexperienced database designers to represent invalid values

13

Two-way equi-join

- Q: $R(A, B) \bowtie S(A, C)$
- Assumption: containment of value sets
  - Every tuple in the “smaller” relation (one with fewer distinct values for the join attribute) joins with some tuple in the other relation
  - That is, if $| \pi_A R | \le | \pi_A S |$ then $\pi_A R \subseteq \pi_A S$
  - Certainly not true in general
  - But holds in the common case of foreign key joins
- $| Q | \approx | R | \cdot | S | / \max(| \pi_A R |, | \pi_A S |)$
  - Selectivity factor of $R.A = S.A$ is $1 / \max(| \pi_A R |, | \pi_A S |)$

14

Multiway equi-join (slide 1)

- Q: $R(A, B) \bowtie S(B, C) \bowtie T(C, D)$
- What is the number of distinct C values in the join of R and S?
- Assumption: preservation of value sets
  - A non-join attribute does not lose values from its set of possible values
  - That is, if A is in R but not S, then $\pi_A (R \bowtie S) = \pi_A R$
  - Certainly not true in general
  - But holds in the common case of foreign key joins

15

Multiway equi-join (slide 2)

- Q: $R(A, B) \bowtie S(B, C) \bowtie T(C, D)$
- Start with the product of relation sizes
  - $| \mathrm{R} | \cdot | \mathrm{S} | \cdot | \mathrm{T} |$
- Reduce the total size by the selectivity factor of each join predicate
  - $R.B = S.B$: $1 / \max(| \pi_B R |, | \pi_B S |)$
  - $S.C = T.C$: $1 / \max(| \pi_C S |, | \pi_C T |)$
  - $| \mathrm{Q} | \approx (| \mathrm{R} | \cdot | \mathrm{S} | \cdot | \mathrm{T} |) / (\max(| \pi_B R |, | \pi_B S |) \cdot \max(| \pi_C S |, | \pi_C T |))$

16

Multiway equi-join (slide 3)

- A slightly more complicated example
  Q: $R(A, B) \bowtie S(A, C) \bowtie T(A, D)$
  - A is common to all three tables
  - $R.A = S.A$ AND $R.A = T.A$ AND $S.A = T.A$
  - Suppose $| \pi_A R |$ is the smallest; consider only $R.A = S.A$ and $R.A = T.A$ ($S.A = T.A$ is implied)
    - $| \mathrm{Q} | \approx (| \mathrm{R} | \cdot | \mathrm{S} | \cdot | \mathrm{T} |) / (\max(| \pi_A R |, | \pi_A S |) \cdot \max(| \pi_A R |, | \pi_A T |))$ $= (| \mathrm{R} | \cdot | \mathrm{S} | \cdot | \mathrm{T} |) / (| \pi_A S | \cdot | \pi_A T |)$
- In general, if a join attribute A appears in multiple tables $R_1, R_2, \ldots, R_n$
  - Divide the total size by the all but the least of $| \pi_A R_i |$

17

Summary

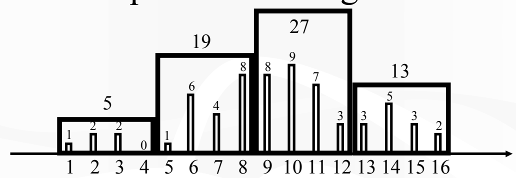
- Using similar ideas, we can estimate the size of projection, duplicate elimination, union, difference, aggregation (with grouping)
- Lots of assumptions and very rough estimation
  - Accurate estimate is not needed
  - Fine if we overestimate or underestimate consistently
  - Sometimes may lead to very nasty optimizer “hints”
    - SELECT * FROM Student WHERE GPA > 3.9;
    - SELECT * FROM Student WHERE GPA > 3.9 AND GPA > 3.9;

18

Better estimation using histograms

- Motivation
  - $|R|$, $|\pi_A R|$, high($R.A$), low($R.A$)
    - Too little information
  - Actual distribution of $R.A$: (v_1, f_1), (v_2, f_2), …, (v_n, f_n)
    - f_i is frequency of v_i, or the number of times v_i appears as $R.A$
    - Too much information
  - Anything in between?
- Idea
  - Partition the domain of $R.A$ into buckets
  - Store a small summary of the distribution within each bucket
  - Number of buckets is the "knob" that controls the resolution

Equi-width histogram



- Divide the domain into B buckets of equal width
- Store the bucket boundaries and the sum of frequencies of the values within each bucket

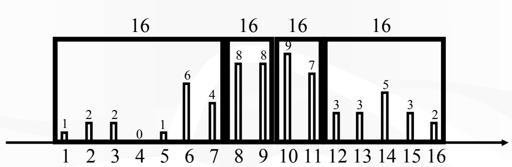
Construction and maintenance

- Construction
  - If high($R.A$) and low($R.A$) are known, use one pass over R to construct an accurate equi-width histogram
    - Keep a running count for each bucket
  - If scanning is unacceptable, use sampling
    - Construct a histogram on R_{sample}, and scale frequencies by $|R| / |R_{sample}|$
- Maintenance
  - Incremental maintenance: for each update on R, increment/decrement the corresponding bucket frequencies
  - Periodical recomputation: because distribution changes slowly

Using an equi-width histogram

- Q: $\sigma_{A=5}\, R$
  - 5 is in bucket [5, 8] (with 19 tuples)
  - Assume uniform distribution within the bucket
  - $|Q| \approx 19/4 \approx 5$ ($|Q| = 1$, actually)
- Q: $\sigma_{A \geq 7 \text{ AND } A \leq 16}\, R$
  - [7, 16] covers [9, 12] (27) and [13, 16] (13)
  - [7, 16] partially covers [5, 8] (19)
  - $|Q| \approx 19/2 + 27 + 13 \approx 50$ ($|Q| = 52$, actually)

Equi-height histogram



- Divide the domain into B buckets with roughly the same number of tuples in each bucket
- Store this number and the bucket boundaries
- Intuition: high frequencies are more important than low frequencies

Construction and maintenance

- Construction
  - Sort all $R.A$ values, and then take equally spaced splits
    - Example: 1 2 2 3 4 7 8 9 10 10 10 10 11 11 12 12 14 16 …
  - Sampling also works
- Maintenance
  - Incremental maintenance
    - Merge adjacent buckets with small counts
    - Split any bucket with a large count
      - Select the median value to split
      - Need a sample of the values within this bucket to work well
  - Periodic recomputation also works

Using an equi-height histogram

- Q: $\sigma_{A=5}\ R$
  - 5 is in bucket [1, 7] (16)
  - Assume uniform distribution within the bucket
  - $|\ Q\ | \approx 16/7 \approx 2$ ($|\ Q\ | = 1$, actually)
- Q: $\sigma_{A \geq 7 \text{ AND } A \leq 16}\ R$
  - [7, 16] covers [8, 9], [10, 11], [12, 16] (all with 16)
  - [7, 16] partially covers [1, 7] (16)
  - $|\ Q\ | \approx 16/7 + 16 + 16 + 16 \approx 50$ ($|\ Q\ | = 52$, actually)

25

Histogram tricks

- Store the number of distinct values in each bucket
  - To get rid of the effects of the values with 0 frequency
    - These values tend to cause underestimation
- Compressed histogram
  - Store (v_i, f_i) pairs explicitly if f_i is high
  - For other values, use an equi-width or equi-height histogram

26

More histograms

- V-optimal histogram
  - Avoid putting very different frequencies into the same bucket
  - Partition in a way to minimize $\sum_i VAR_i$, where VAR_i is the frequency variance within bucket i
- MaxDiff histogram
  - Define area to be the product of the frequency of a value and its “spread” (the difference between this value and the next value with non-zero frequency)
  - Insert bucket boundaries where two adjacent areas differ by large amounts
- More in Poosala et al., SIGMOD 1996

27